

00862.023507



PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:)	
	:	Examiner: Unassigned
HIROKI KITAMURA, ET AL.)	
	:	Group Art Unit: Unassigned
Appln. No.: 10/805,324)	
	:	
Filed: March 22, 2004)	
	:	
For: PRINTING APPARATUS AND)	
PRINTING DATA CONTROL	:	
METHOD)	July 6, 2004

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUBMISSION OF PRIORITY DOCUMENT

Sir:

In support of Applicants' claim for priority under 35 U.S.C. § 119, enclosed
is a certified copy of the following Japanese application:

No. 2003-081060 filed March 24, 2003.

Applicants' undersigned attorney may be reached in our Washington, D.C. office by telephone at (202) 530-1010. All correspondence should continue to be directed to our below-listed address.

Respectfully submitted,

A handwritten signature in black ink, appearing to read "Mark A. Williamson", is written over a horizontal line.

Attorney for Applicants

Mark A. Williamson

Registration No. 33,628

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

MAW\tnr

DC_MAIN 169242v1

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 3 月 2 4 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 0 8 1 0 6 0
Application Number:
[ST. 10/C]: [J P 2 0 0 3 - 0 8 1 0 6 0]

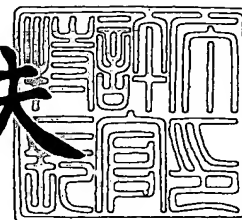
出 願 人 キヤノン株式会社
Applicant(s):

10/805,324

2 0 0 4 年 4 月 1 2 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



【書類名】 特許願

【整理番号】 226141

【提出日】 平成15年 3月24日

【あて先】 特許庁長官殿

【国際特許分類】 B41J 2/00
G06F 3/00

【発明の名称】 記録装置

【請求項の数】 1

【発明者】

【住所又は居所】 東京都大田区下丸子3丁目30番2号 キヤノン株式会社
社内

【氏名】 北村 宏記

【発明者】

【住所又は居所】 東京都大田区下丸子3丁目30番2号 キヤノン株式会社
社内

【氏名】 石川 尚

【発明者】

【住所又は居所】 東京都大田区下丸子3丁目30番2号 キヤノン株式会社
社内

【氏名】 川村 興二

【特許出願人】

【識別番号】 000001007

【氏名又は名称】 キヤノン株式会社

【代理人】

【識別番号】 100076428

【弁理士】

【氏名又は名称】 大塚 康德

【電話番号】 03-5276-3241

【選任した代理人】

【識別番号】 100112508

【弁理士】

【氏名又は名称】 高柳 司郎

【電話番号】 03-5276-3241

【選任した代理人】

【識別番号】 100115071

【弁理士】

【氏名又は名称】 大塚 康弘

【電話番号】 03-5276-3241

【選任した代理人】

【識別番号】 100116894

【弁理士】

【氏名又は名称】 木村 秀二

【電話番号】 03-5276-3241

【手数料の表示】

【予納台帳番号】 003458

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0102485

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 記録装置

【特許請求の範囲】

【請求項 1】 所定方向に配列された複数の記録素子を有する記録ヘッドを搭載したキャリッジを、前記記録素子の配列方向と交差する方向に記録媒体上で走査させて記録を行う記録装置であって、

ラスト形式の記録データを格納する記録データメモリと、

各記録素子に対応して格納領域を有し、前記記録データメモリに格納された記録データを格納するバッファメモリと、

前記記録ヘッドの構成に関する情報を格納するヘッドパラメータ部と、

前記ヘッドパラメータ部に格納された情報に応じて、前記記録データメモリに格納された記録データの前記バッファメモリへの転送順序と、前記バッファメモリに格納された記録データの読み出し順序とを制御するバッファ制御部と、を備えることを特徴とする記録装置。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は記録装置に関し、特に、所定方向に配列された複数の記録素子を有する記録ヘッドを搭載したキャリッジを、記録素子の配列方向と交差する方向に記録媒体上で走査させて記録を行う記録装置における記録データを処理する技術に関する。

【0 0 0 2】

【従来の技術】

例えばワードプロセッサ、パーソナルコンピュータ、ファクシミリ等に於ける情報出力装置として、所望される文字や画像等の情報を用紙やフィルム等シート状の記録媒体に記録を行うプリンタが広く使用されている。

【0 0 0 3】

プリンタの記録方式としては様々な方式が知られているが、用紙等の記録媒体に非接触記録が可能である、カラー化が容易である、静粛性に富む、等の理由で

インクジェット方式が近年特に注目されており、又その構成としては所望される記録情報に応じてインクを吐出する記録ヘッドを装着すると共に用紙等の記録媒体の搬送方向と交差する方向に往復走査しながら記録を行なうシリアル記録方式が安価で小型化が容易などの点から一般的に広く用いられている。

【0 0 0 4】

このようなインクジェットプリンタの記録ヘッドは、1回の走査で記録する領域を大きくして記録速度を向上すべく、記録媒体の搬送方向とほぼ同じ方向に多数の記録素子（ノズル）が配列され、カラー記録及び／又は多階調記録のために、使用する複数のインクに対応して記録素子の列あるいは記録ヘッドを複数備える構成が一般的である。

【0 0 0 5】

通常、パーソナルコンピュータ等のホスト機器から画像データは、走査方向（水平方向）に展開された1ライン単位のラスタデータとして送信され、これを受信したプリンタでは、記録素子の配列方向（垂直方向）にあわせて、画像データを並び替えるH－V変換が行われている。

【0 0 0 6】

インクジェット方式のプリンタで高速でかつ安定したインクの吐出を行うために、全てのノズルから同時にインクを吐出させるのではなく、ある特定の間隔で配列された複数のノズルを同時に吐出させるように制御する分散吐出制御方式が用いられている。

【0 0 0 7】

この分散吐出制御方式を用いたインクジェット方式のプリンタにおいて、カラー記録等で複数のインクを用いて記録を行う場合、使用するインク毎に、データ管理および分散吐出の駆動制御を行っている。

【0 0 0 8】

例えば、使用するインクとして、減色混色の3原色であるイエロー、マゼンタ、シアンの3色のインクを用いるカラープリンタでは、各色の記録ヘッドの構成（ノズル数、記録ヘッドの位置関係等）に応じて制御回路を構成しており、記録ヘッドの構成が異なる場合には、制御回路をその構成に応じて変更している。

【0009】

【発明が解決しようとする課題】

近年、インクジェットプリンタは様々な用途で利用される機会が増えているが、記録速度の向上や高画質化に対する要望も強い。このため、記録ヘッドの構成は、用途や目的に応じて非常に多岐にわたっている。

【0010】

開発を容易にし、かつ低価格化を可能とするためには、様々な記録ヘッドの構成に対応できる制御回路を設計するのが望ましい。

【0011】

具体的に制御回路は、記録ヘッド毎にバッファメモリを設けてヘッドへ転送するデータの生成を行うように設計されるが、このように記録ヘッドの構成が様々なであると、制御回路の構成は、使用する記録ヘッドの構成毎に異なる回路構成となり、他の構成の記録ヘッドに対して利用することが不可能である。

【0012】

例えば、最も簡単な構成として、1回の走査で記録されるデータを格納するバッファメモリを記録ヘッド（インク）毎に設ける構成では、バッファメモリの容量が記録に用いる記録媒体のサイズ及び解像度に応じて変化する。処理の高速化のために、このバッファメモリを超大規模集積回路（VLSI）に形成する場合、バッファメモリがVLSI上のほとんどの領域を占めることとなる。たとえバッファメモリのサイズを小さくしたとしても、様々な構成の記録ヘッドに対応して複数種類ものバッファメモリを形成すると、メモリが占める物理的領域が増大することになり効率的ではない。

【0013】

以上のように、記録ヘッドの制御回路及び該制御回路を含む大規模な論理回路を様々な構成の記録ヘッドに対して共通とすることは困難であり、このため、記録ヘッドの制御回路は、開発の効率化及び装置の低価格化を達成するのに障害となっている。

【0014】

本発明は以上のような状況に鑑みてなされたものであり、シリアル記録方式の

記録装置において、様々な構成の記録ヘッドに対して共通の制御回路を使用可能とすることを目的とする。

【0015】

【課題を解決するための手段】

上記目的を達成する本発明の一態様としての記録装置は、所定方向に配列された複数の記録素子を有する記録ヘッドを搭載したキャリッジを、前記記録素子の配列方向と交差する方向に記録媒体上で走査させて記録を行う記録装置であって、

ラスタ形式の記録データを格納する記録データメモリと、

各記録素子に対応して格納領域を有し、前記記録データメモリに格納された記録データを格納するバッファメモリと、

前記記録ヘッドの構成に関する情報を格納するヘッドパラメータ部と、

前記ヘッドパラメータ部に格納された情報に応じて、前記記録データメモリに格納された記録データの前記バッファメモリへの転送順序と、前記バッファメモリに格納された記録データの読み出し順序とを制御するバッファ制御部と、を備えている。

【0016】

すなわち、本発明では、所定方向に配列された複数の記録素子を有する記録ヘッドを搭載したキャリッジを、記録素子の配列方向と交差する方向に記録媒体上で走査させて記録を行う記録装置において、送信された画像情報を印刷に使用するデータ処理をしラスタ形式化された記録データ、あるいは、ラスタ形式で送信された記録データを、記録データメモリに格納し、記録データメモリに格納された記録データを各記録素子に対応して格納領域を有するバッファメモリに格納する構成において、記録ヘッドの構成に関する情報を格納するヘッドパラメータ部を設け、ヘッドパラメータ部に格納された情報に応じて、記録データメモリに格納された記録データのバッファメモリへの転送順序と、バッファメモリに格納された記録データの読み出し順序とを制御する。

【0017】

このようにすると、様々な構成の記録ヘッドに対して共通の制御回路を使用す

ることが可能となり、制御回路の共通化によるコストダウン、並びに装置の設計期間を短縮することができる。

【0 0 1 8】

なお、本発明は上記の記録装置としての態様以外にも、記録装置における記録データの制御方法、該記録データの制御方法を実現するコンピュータプログラム、該コンピュータプログラムを格納する記憶媒体の態様でも実現可能である。

【0 0 1 9】

【発明の実施の形態】

上記のように本発明は、所定方向に配列された複数の記録素子を有する記録ヘッドを搭載したキャリッジを、記録素子の配列方向と交差する方向に記録媒体上で走査させて記録を行う記録装置であって、ラスタ形式の記録データを格納する記録データメモリと、各記録素子に対応して格納領域を有し、記録データメモリに格納された記録データを格納するバッファメモリと、記録ヘッドの構成に関する情報を格納するヘッドパラメータ部と、ヘッドパラメータ部に格納された情報に応じて、記録データメモリに格納された記録データのバッファメモリへの転送順序と、バッファメモリに格納された記録データの読み出し順序とを制御するバッファ制御部と、を備える記録装置であるが、如何に説明する実施形態は次のような特徴をも備えている。

【0 0 2 0】

バッファメモリは各記録素子に対応した格納領域をそれぞれ有する2つのバッファを含んでおり、バッファ制御部は、格納された記録データを読み出すバッファと記録データメモリからの記録データ転送先のバッファとを、2つのバッファ間で交互に切り替える。

【0 0 2 1】

記録に使用する複数の色に対応して記録ヘッドを複数備えており、ヘッドパラメータ部は、記録ヘッドの数、各記録ヘッドの記録素子の数、及び記録ヘッドの配列方向に関する情報を格納する。

【0 0 2 2】

バッファメモリが、全ての記録素子に対応した格納領域よりも容量が大きい。

【0023】

記録ヘッドは、インクを吐出して記録を行うインクジェット記録ヘッドである。

【0024】

なお、インクジェットの方式としては様々な方式が考えられるが、記録ヘッドが熱エネルギーを利用してインクを吐出すべく、インクに与える熱エネルギーを発生するための熱エネルギー変換体を備えているのが好ましい。

【0025】

以下添付図面を参照して本発明の好適な実施形態について詳細に説明する。

【0026】

なお、以下に説明する実施形態では、インクジェット記録方式を用いた記録装置としてプリンタを例に挙げ説明する。

【0027】

本明細書において、「記録」（「プリント」という場合もある）とは、文字、図形等有意の情報を形成する場合のみならず、有意無意を問わず、また人間が視覚で知覚し得るように顕在化したものであるか否かを問わず、広く記録媒体上に画像、模様、パターン等を形成する、または媒体の加工を行う場合も表すものとする。

【0028】

また、「記録媒体」とは、一般的な記録装置で用いられる紙のみならず、広く、布、プラスチック・フィルム、金属板、ガラス、セラミックス、木材、皮革等、インクを受容可能なものも表すものとする。

【0029】

さらに、「インク」（「液体」と言う場合もある）とは、上記「記録（プリント）」の定義と同様広く解釈されるべきもので、記録媒体上に付与されることによって、画像、模様、パターン等の形成または記録媒体の加工、或いはインクの処理（例えば記録媒体に付与されるインク中の色剤の凝固または不溶化）に供され得る液体を表すものとする。

【0030】

図7は、本発明の代表的な実施の形態であるインクジェットプリンタ I J R A の構成の概要を示す外観斜視図である。図7において、駆動モータ 5013 の正逆回転に連動して駆動力伝達ギア 5009 ~ 5011 を介して回転するリードスクリュー 5005 の螺旋溝 5004 に対して係合するキャリッジ H C はピン（不図示）を有し、ガイドレール 5003 に支持されて矢印 a, b 方向を往復移動する。キャリッジ H C には、記録ヘッド 10 とインクタンク I T とを内蔵した一体型インクジェットカートリッジ I J C が搭載されている。

【0031】

5002 は紙押え板であり、キャリッジ H C の移動方向に互って記録用紙 P をプラテン 5000 に対して押圧する。5007, 5008 はフォトカプラで、キャリッジのレバー 5006 のこの域での存在を確認して、モータ 5013 の回転方向切り換え等を行うためのホームポジション検知器である。

【0032】

5016 は記録ヘッド 10 の前面をキャップするキャップ部材 5022 を支持する部材で、5015 はこのキャップ内を吸引する吸引器で、キャップ内開口 5023 を介して記録ヘッドの吸引回復を行う。5017 はクリーニングブレードで、5019 はこのブレードを前後方向に移動可能にする部材であり、本体支持板 5018 にこれらが支持されている。なお、ブレードは、この形態に限らず周知のクリーニングブレードが本例に適用できることは言うまでもない。

【0033】

又、5021 は、吸引回復の動作を開始するためのレバーで、キャリッジと係合するカム 5020 の移動に伴って移動し、駆動モータからの駆動力がクラッチ切り換え等の公知の伝達機構で移動制御される。

【0034】

これらのキャッピング、クリーニング、吸引回復は、キャリッジがホームポジション側の領域に来た時にリードスクリュー 5005 の作用によってそれらの対応位置で所望の処理が行えるように構成されているが、周知のタイミングで所望の動作を行うようにすれば、本例にはいずれも適用できる。

【0035】

図2は、本実施形態のインクジェットプリンタの記録に関する制御構成のブロック図である。

【0036】

図示されたように、本実施形態のプリンタの記録に関する制御ブロックは、記録に関する制御を受け持つ記録制御部1、装置全体を制御するCPU5、記録データを格納する記録データ用メモリ6、記録データ用メモリ6へのアクセスを制御するメモリ制御部12、インクを吐出性能維持回復して記録を行う記録ヘッド10、及び記録ヘッドを搭載したキャリッジ駆動機構並びに記録媒体搬送機構を含む記録機構駆動部9を含んでいる。

【0037】

記録制御部1は、バッファメモリ7、バッファメモリへのアクセスを制御するバッファ制御部2、記録データを記録ヘッドに合わせた形式に変換する記録データ変換部3、記録ヘッドの構成に関する情報を格納するヘッドパラメータ部4、記録機構駆動部からの情報に基づきタイミング信号を制御する記録タイミング制御部8を含んでいる。

【0038】

本実施形態における記録の際の処理の概要は、CPU5が不図示のCPUの制御プログラムを格納したメモリから、記録制御用のプログラムを読み出し、不図示の電気回路により記録機構駆動部9に動作用の信号を供給し、この信号により動作する記録機構駆動部9から位置情報などを受け取り、記録データ用メモリ6に格納されている内容を記録制御部1内の記録データ変換部で変換して記録ヘッド10にデータを転送し、記録を実行する。

【0039】

記録ヘッド10に送られるデータは、記録ヘッドを移動させつつ記録を実行するという構造を考慮して、1本のデータラインによりデータを転送し、記録ヘッド内に設けられたシフトレジスタ及びラッチを含むロジック回路により、シリアルに転送されたデータをパラレルに変換して各記録素子の駆動回路に供給する。

【0040】

図3及び図4は、本実施形態のプリンタに搭載する記録ヘッドの代表的ノズル

構成を示す図である。ここではいずれも 3 種類のインクを使用する場合の例を示しているが、使用するインクの種類の数は 3 以上であってもよいのはもちろんである。

【0041】

図 3 は、それぞれのインクに対して複数（0～n）のノズルを有する 3 つの記録ヘッド 301～303 が 1 直線上に配置された構成である。3 つの記録ヘッドのノズルが並んでいる方向と交差する方向に記録ヘッドが移動（走査）されて記録が行われ、各走査の終了後、記録媒体のノズルの配列方向とほぼ同じ方向への搬送を繰り返すことにより、1 枚の記録媒体への記録を行う。

【0042】

この例のように、3 種類のインクを使用する場合には、各走査終了後に 1 つの記録ヘッドの長さ（ノズル列の長さ）に相当する距離だけ記録媒体を搬送することにより、3 種類のインクを同じ位置（画素）に対して選択的に吐出することが可能となり、3 種類のインクによる記録が可能となる。

【0043】

より具体的に説明すると、記録ヘッド 301 がシアンインク、302 がマゼンタインク、303 がイエローインクをそれぞれ吐出すると想定すると、1 回目の走査でノズル C0-0 によって記録された画素に対して、2 回目の走査ではノズル C1-0 で記録紙、3 回目の走査ではノズル C2-0 で記録を行うことにより、3 色のインクによる混色カラー記録が可能となる。

【0044】

図 4 は、それぞれのインクに対して複数（0～n）のノズルを有する 3 つの記録ヘッド（a）、（b）及び（c）が並列に配置された構成である。この場合にも、走査方向及び記録媒体の搬送方向は図 3 の場合と同様であり、各走査の終了後、記録媒体のノズルの配列方向とほぼ同じ方向への、1 つの記録ヘッドの長さ（ノズル列の長さ）に相当する距離の搬送を繰り返すことにより、1 枚の記録媒体への記録を行う。

【0045】

図 3 の構成と図 4 の構成を比較すると、図 3 の構成では 3 つの記録ヘッドのノ

ズルが1直線上に並ぶように直列に配置されているが、図4の構成では3つの記録ヘッドのノズルが同じ水平位置となるように並列に配置されている点で異なっている。

【0046】

図4に示すような構成では、1回の走査で同じ画素に対して3種類のインクを選択的に吐出できる、すなわち、1パスでカラー記録が可能となるが、3つの記録ヘッドをそれぞれ異なったタイミングで駆動する制御が必要となる。

【0047】

記録ヘッドの構成としては、図3及び図4に示した以外にも様々な構成が知られており、どのような構成とするのかは、装置の価格や大きさ、目標とする記録速度や対象とする記録媒体のサイズなどの様々な要因によって決定される。

【0048】

上述のように従来は、記録ヘッド毎にバッファメモリを設けていたため、記録ヘッドの構成が異なるとバッファメモリの構成も異なっていたが、本実施形態ではノズル毎にバッファメモリを割当てて、記録ヘッドの構成に関らず、同じ構成のバッファメモリを使用可能とする。

【0049】

本実施形態においては、図1に示すバッファメモリ7の構成例のように、記録データ格納用のバッファメモリとして全てのノズルに対するバッファメモリを一つにまとめたバッファメモリ7として構成し、ヘッドパラメータ部4に格納された情報（記録ヘッドの数、各記録ヘッドのノズル数により、記録ヘッドの配列方向が直列（図3）であるかあるいは並列（図4）であるかを意識することなく、各ノズルに対するデータの書込み／読み出しを制御することにより、図3又は図4に示したいずれのヘッド構成であっても同じ論理回路によりノズルに対するデータ生成を行うことを可能にする。

【0050】

なお、図2のブロック図においてバッファメモリ7が7-1と7-2の2つの部分に分かれているのは、バッファメモリ7に保存された記録データを記録データ変換部3にバッファ制御部2を経由して転送する際に、一方でバッファ制御部

2にデータを転送し、他方で記録データ用メモリ6からデータを受信するようにして、記録ヘッドへのデータ転送をより高速に実行するためである。後述するように、7-1及び7-2は、データ転送及びデータ受信をそれぞれ交互に行う。

【0051】

以下、本実施形態における記録ヘッドへのデータ転送の動作について説明する。

【0052】

ヘッドパラメータ部4には、記録ヘッドの構成に関する情報が格納されている。図9にパラメータ部に格納される情報の例を示す。具体的には、記録ヘッドの数(c_num)、各記録ヘッドのノズルの数(n_num)、及び次のノズルの記録データが記録メモリーに記録されている位置を表す情報($n1_diff$)、次の記録ヘッドの記録データが記録メモリーに記録されている位置を表す情報($c0_diff$ 、 $c1_diff$ 、 $c2_diff$)、記録データの先頭が記録されている位置を表す情報(bm_strt_adr)が格納されている。

【0053】

この記録ヘッドの数 c_num と各記録ヘッドのノズル数 n_num を用いたバッファ7におけるデータの管理を図1のバッファ構成例を用いて説明する。図1の例で使用する記録ヘッドは、C0、C1、C2の3つであり、 c_num は3となる。また、3つの記録ヘッドC0、C1、C2のノズル数 n_num をそれぞれ n 、 m 、 l 、バッファメモリ7の記録データ格納用の素子(アレイ)の数を N ($n+m+l < N$)として説明する。バッファメモリ7のアドレスは、アレイを指定するアドレスであり、0から $N-1$ の値をとる。

【0054】

バッファメモリ7に格納する順番をC0、C1、C2とすると、バッファメモリ7のアドレス0から $n-1$ までは記録ヘッドC0の0から $n-1$ までの n 個のノズルに対する記録データ、バッファメモリ7のアドレス n から $n+m-1$ までは記録ヘッドC1の0から $m-1$ までの m 個のノズルに対する記録データ、バッファメモリ7のアドレス $n+m$ から $n+m+l-1$ までは記録ヘッドC2の0から $l-1$ までの l 個のノズルに対する記録データがそれぞれ格納される。バッフ

ァメモリ 7 のアドレス $n+m+1$ から $N-1$ までは、本例では使用されない。

【0055】

バッファメモリ 7 の記録データ格納用の単位となるアレイのサイズ（データ幅、あるいはビット数）は、システムの構成条件により決定される。この条件は、記録制御部 1 において 2 個あるバッファメモリを前述したように交互に切り替えて使用するとき、一方のバッファを記録データの生成に使用している間に他方のバッファに次の記録データを格納するのに要する時間から決定されるか、もしくは、システムで使用されているデータ幅（8 ビット／16 ビット／32 ビットなど）を考慮して決定される。

【0056】

以上のような構成におけるバッファ制御処理の流れを、図 5 のフローチャートと図 6 のタイミングチャートを参照して説明する。

【0057】

バッファ制御部 2 によるバッファメモリ 7 への書き込みは、CPU 5 からの指示により開始する。CPU 5 は、記録開始に先立ち、記録に必要な記録データを記録データ用メモリ 6 に格納する。バッファ制御部 2 はバッファメモリ 7 への書き込み時には、必要なデータを記録データ用メモリ 6 から読み出す。メモリ制御部 12 は、CPU 5 から受信したデータを格納すると共に、バッファ制御部 2 からの要求に基づき記録データをバッファ制御部 2 に転送する 2 つの動作を行う。

【0058】

ここで、図 6 のタイミングチャートに示された信号について説明すると、位置パルス（a）は記録機構駆動部 9 のエンコーダ等から発生される信号であり、この信号に基づいて、位置カウント（b）が生成される。列パルス（d）は記録タイミング制御部 8 で位置パルス（a）から記録する列に対応して生成するパルスであり、この列パルスを複数に分割して記録ヘッドを駆動する際の列分割パルス（e）を生成し、列分割パルスからバッファ制御部 2 へのデータ要求パルス（f）を生成する。また、（g）、（h）及び（i）は、（d）、（e）及び（f）の信号をそれぞれ時間軸方向に拡大して示したものである。

【0059】

バッファ制御部 2 は CPU 5 からの開始指示により、まず、記録データ用メモリのアドレス rd_addr は初期値 bm_strt_addr に設定され、バッファメモリ 7-1 に対して、図 1 のバッファの構成に基づいて、記録データ用メモリ 6 の rd_addr から読み出した記録データをバッファメモリ 7-1 のアドレス 0 に格納し、 $bm_strt_addr + nl_diff$ から読み出した記録データをバッファメモリ 7-1 のアドレス 1 に格納し、 $bm_strt_addr + nl_diff \times (n-1)$ の記録データをアドレス $n-1$ に格納する。 rd_addr とバッファメモリのアドレスとの関係は、 k をバッファメモリのアドレスとすると、

$$\begin{aligned} c0 : rd_addr &= bm_strt_addr + nl_diff \times (k-1) \\ c1 : rd_addr &= bm_strt_addr + nl_diff \times (k-2) \\ &+ c0_diff \\ c2 : rd_addr &= bm_strt_addr + nl_diff \times (k-3) \\ &+ c0_diff + c1_diff \end{aligned}$$

となる。

【0060】

上記の関係で逐次データをバッファメモリ 7-1 に記録データを格納する。 rd_addr は前記関係式で常に計算するのではなく前回の結果に対して演算を繰り返すことで達成する。(ステップ S501)。このバッファメモリ 7-1 へのデータ格納終了後、後続する記録データを同様にして、前回の rd_addr に $c1_diff$ を足した値をバッファメモリ 7-2 のアドレス 0 のアドレスとして記録データ用メモリ 6 から読み出してバッファメモリ 7-2 に格納し、以下上記処理を繰り返し格納する (ステップ S502)。

【0061】

バッファメモリを転送待機状態とした (ステップ S503) 後、記録動作が開始されタのを検知したら (ステップ S504)、記録機構駆動部 9 から記録タイミング制御部 8 に送られる、図 6 に示す位置パルス (a) および位置カウント (b) などの位置情報に応じて生成される (f) 及び (i) に示すデータ要求パルスの受信により、記録タイミング制御部 8 から記録データ変換部 3 へのデータ転

送がバッファ制御部 2 に要求されたのを検知して（ステップ S 5 0 5）、バッファ制御部 2 によるバッファメモリ 7-1 からのデータ転送が実行される。

【0062】

また、バッファ制御部 2 は、バッファ制御部 2 から記録データ変換部 3 に送信したデータが有効であることを示す信号を供給し、記録データ変換部 3 は、バッファ制御部 2 から受信した記録データを変換して記録ヘッドに供給する。

【0063】

バッファメモリ 7-1 からバッファ制御部 2 へのデータ転送であると判定されたら（ステップ S 5 0 6）、バッファメモリの記憶単位（アレイ）毎に、図 6 のタイミングチャートの（f）及び（i）に示すデータ要求パルスに対応して行われる（ステップ S 5 0 7）。データ要求パルスを受信する度にこのアレイ毎のデータ転送を繰り返し、バッファメモリ 7-1 に格納された分だけデータ要求パルスが出力されたときに、バッファメモリ 7-1 のデータが全部使用されたと判定する（ステップ S 5 0 8）。この時、転送待機状態をバッファメモリ 7-1 からバッファメモリ 7-2 に移行させ（ステップ S 5 0 9）、記録データ用メモリに格納された記録データのバッファメモリ 7-1 への転送を開始する（ステップ S 5 1 0）。

【0064】

図 6 の（a）～（c）の信号から記録終了か否かを判定し（ステップ S 5 1 5）、記録終了でないと判定された場合には、次の列パルスに対するデータ要求に対して、バッファ 7-2 からのデータを記録データ変換部 3 へ転送すべく、ステップ S 5 0 5 に戻る。

【0065】

このとき、今回はバッファメモリ 7-2 のデータ転送であるので、ステップ S 5 0 6 から S 5 1 1 へ進み、上記と同様にデータ要求パルスを受信する度にこのアレイ毎のデータ転送を繰り返し、バッファメモリ 7-2 に格納された分だけデータ要求パルスが出力されたときに、バッファメモリ 7-2 のデータが全部使用されたと判定する（ステップ S 5 1 2）。この時、転送待機状態をバッファメモリ 7-2 からバッファメモリ 7-1 に移行させ（ステップ S 5 1 3）、記録デー

タ用メモリに格納された記録データのバッファメモリ 7-2 への転送を開始する (ステップ S 5 1 4)。

【0066】

そして、ステップ S 5 1 5 で記録終了と判定された場合、バッファ制御処理を終了する。

【0067】

<VHDLによる具体例>

以下、上記実施形態の構成を、ハードウェア記述言語を用いた場合の記述例について説明する。図 8 A ~ 8 K は、図 2 のブロック図のバッファ制御部 2 と記録データ用メモリ読み出し制御部 1 1 の構成を、ハードウェア記述言語として VHDL を用いた場合の記述例を示している。本記述例は、図 5 のフローチャートに基づいているが、ひとつの処理で図 5 のフローチャート中の複数の処理をかねている。以下、この記述の内容に関して説明する。

【0068】

図 8 A において、entity で始まり end で終わる部分は、このブロック全体の入出力を宣言している部分であり、このうち GENERIC () 内は、このブロックで使用されるデータのサイズを他のシステムに利用するとき最適なサイズに変更して使用することができるようにした変数を宣言している記述であり、本例では、横方向の列の最大値、すなわち、1 回の走査での最大記録位置数を 8 1 9 2 個 (0 ~ 8 1 9 1 個、X__MSB = 1 3)、バッファ制御部 2 から記録データ変換部 3 へのデータの幅が 6 ビット (B__NUM = 5)、外部の記録用データメモリ 6 にアクセスする際のアドレスが 2 2 ビットであり、データ幅が 3 2 ビット (A__LSB = 2) であるシステムに適用したものであることが記述されている。

【0069】

port () 内は、このブロックに対する入力信号と出力信号を定義している部分である。本例では、4 種類の情報を記録するシステムに適用しており、各情報に対して固有に使用される信号は、信号名の頭に、C0_、C1_、C2_、C3_ のいずれかが付加されているかで各情報に対する信号が識別される。さらに、1 列の記録

を行う際に、記録ヘッドのノズルを16のブロックに分割して時分割駆動するものに適用している。

【0070】

図8Bから8Kの、ARCHITECTURE RTL OF 以下で始まりEND RTL;で終わる部分は、上記の実施形態の動作を実行するための論理記述であり、図8Bの部分は、論理記述で使用する定数 (CONSTANT) 及び信号 (SIGNAL) を定義する部分であり、図8C以降が実際の論理である。

【0071】

図8C以降の内容に関して、より詳細に説明する。

【0072】

本例のブロックは、init入力を'1'に設定することで初期化が開始され、init入力が'0'に遷移すると初期化後からの起動が実行される。起動開始後は、801で示す部分の記述により初期化のパルスおよび記録タイミング制御部8からの要求による読み出しの起動パルスが生成される (ステップS501の一部である)。

【0073】

図8Cの801以降から図8Fまでの部分は、バッファメモリからのデータの読み出しを記録ヘッドの構成に合わせて制御するための記述である (S504、S505およびS507、S511の一部、S515)。

【0074】

図8Gに示される部分は、2つのメモリバッファを交互に使用するためにバッファの使用状態を検出するための記述である (ステップS506、S508、S509、S512、S513)。

【0075】

図8Hに示される部分は、メモリバッファから読み出されたデータに対して、記録データ変換部3で記録ヘッドの構成に応じてデータを変換し、要求された構成でデータを生成するための記述である (ステップS507、S511の一部)。

【0076】

図 8 I に示される部分は、2 つのメモリバッファを交互に使用するために、図 8 G に示す記述で検出された状態に応じて、バッファメモリへのデータ書き込みを行うための記述である（ステップ S 5 1 0、S 5 1 4）。

【0 0 7 7】

図 8 J に示される部分は、図 8 I に示される部分でメモリバッファへの書き込みの要求が発生したときに、外部の記録データ用メモリ 6 に対してデータの要求と、その要求に対する応答を検出し、図 8 I のブロックに記録データ用メモリ 6 から有効なデータが供給されていることを通知制御するための記述である（ステップ S 5 0 1 の一部および S 5 1 4、S 5 1 0 のデータ保存部に関する処理である）。

【0 0 7 8】

図 8 K に示される部分は、図 8 J に示す部分で記録データ用メモリ 6 にデータを要求する際に、記録の進展状況に応じて、記録データ用メモリ 6 内の必要となるデータが存在する場所（アドレス）を演算し、要求されるタイミングで演算されたアドレスを生成するための記述である（ステップ S 5 0 1 の一部および S 5 0 2、S 5 0 3、S 5 1 4、S 5 1 0 を実施する）。

【0 0 7 9】

<他の実施形態>

以上本発明をインクジェットプリンタに適用した場合について説明したが、本発明はシリアル方式の記録装置であれば、インクジェット方式以外の他の方式を採用する記録装置にも適用できる。

【0 0 8 0】

なお、本発明は、複数の機器（例えばホストコンピュータ、インターフェース機器、リーダー、プリンタなど）から構成されるシステムに適用しても、一つの機器からなる装置（例えば、複写機、ファクシミリ装置など）に適用してもよい。

【0 0 8 1】

また、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（または CPU や MPU）が記憶媒体に格納

されたプログラムコードを読み出し実行することによっても、達成されることは言うまでもない。

【0082】

この場合、記憶媒体から読み出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。

【0083】

プログラムコードを供給するための記憶媒体としては、例えば、フレキシブルディスク、ハードディスク、光ディスク、光磁気ディスク、CD-ROM、CD-R、磁気テープ、不揮発性のメモリカード、ROMなどを用いることができる。

【0084】

また、コンピュータが読み出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているOS（オペレーティングシステム）などが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【0085】

さらに、記憶媒体から読み出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるCPUなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【0086】

本発明を上記記憶媒体に適用する場合、その記憶媒体には、先に説明した（図7および／または図8A～8Kに示す）フローチャートやハードウェア記述言語の記述に対応するプログラムコードが格納されることになる。

【0087】

【発明の効果】

以上説明したように本発明によれば、様々な構成の記録ヘッドに対して共通の制御回路を使用することが可能となり、制御回路の共通化によるコストダウン、並びに装置の設計期間を短縮することができる。

【図面の簡単な説明】**【図 1】**

本発明の実施形態におけるバッファメモリの構成例を示す図である。

【図 2】

本発明の実施形態の制御構成を示すブロック図である。

【図 3】

記録ヘッドの構成例を説明するための図である。

【図 4】

記録ヘッドの別の構成例を説明するための図である。

【図 5】

実施形態における記録データのバッファ処理を示すフローチャートである。

【図 6】

実施形態における信号の変化を示すタイミングチャートである。

【図 7】

本発明の好適な実施形態としてのインクジェットプリンタの概略構成を示す外観図である。

【図 8 A】、

【図 8 B】、

【図 8 C】、

【図 8 D】、

【図 8 E】、

【図 8 F】、

【図 8 G】、

【図 8 H】、

【図 8 I】、

【図 8 J】、

【図 8 K】

ハードウェア記述言語による記述の例を示す図である。

【図 9】

ヘッドパラメータ部に格納される情報の例を示す図である。

【符号の説明】

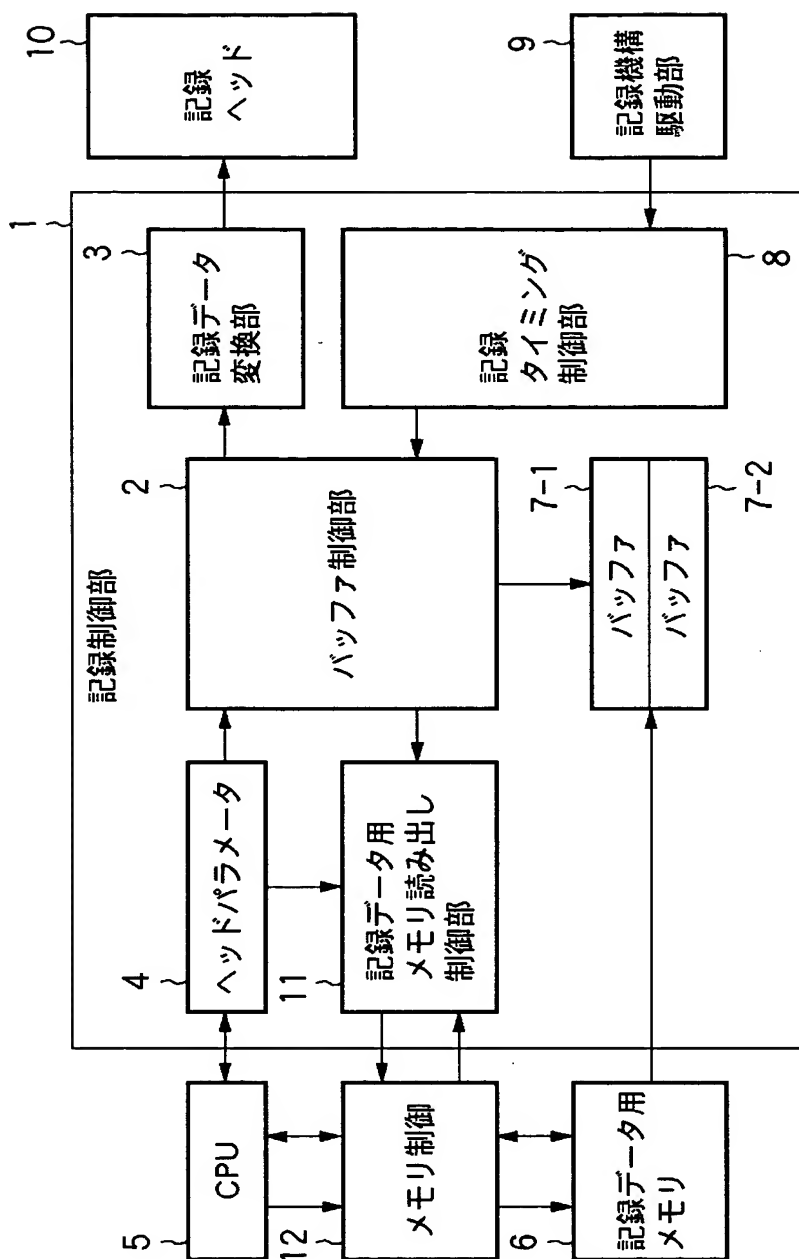
- 1 記録制御部
- 2 バッファ制御部
- 3 記録データ変化部
- 4 ヘッドパラメータ部
- 5 CPU
- 6 記録データ用メモリ
- 7-1 第1のバッファメモリ
- 7-2 第2のバッファメモリ
- 8 記録タイミング制御部
- 9 記録機構駆動部
- 10 記録ヘッド
- 11 記録データ用メモリ読み出し制御部
- 12 メモリ制御部

【書類名】 図面

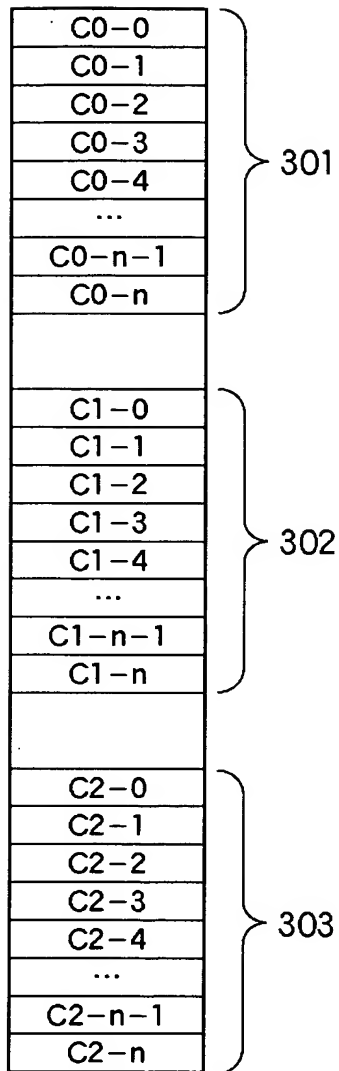
【図 1】

アドレス	データ
0	C0-0
1	C0-1
2	C0-2
3	C0-3
4	C0-4
...	...
n-2	C0-n-2
n-1	C0-n-1
n	C1-0
n+1	C1-1
n+2	C1-2
n+3	C1-3
n+4	C1-4
...	...
n+m-2	C1-m-2
n+m-1	C1-m-1
n+m	C2-0
n+m+1	C2-1
n+m+2	C2-2
n+m+3	C2-3
n+m+4	C2-4
...	...
n+m+l-2	C2-l-2
n+m+l-1	C2-l-1
n+m+l	Blank
n+m+l+1	Blank
n+m+l+2	Blank
n+m+l+3	Blank
n+m+l+4	Blank
...	...
N-2	Blank
N-1	Blank

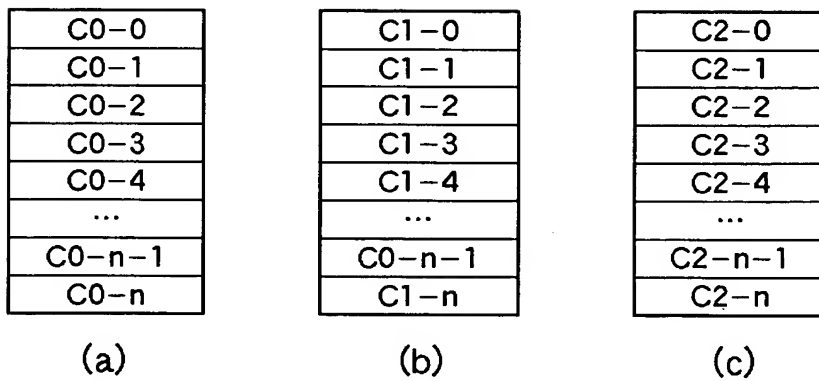
【図 2】



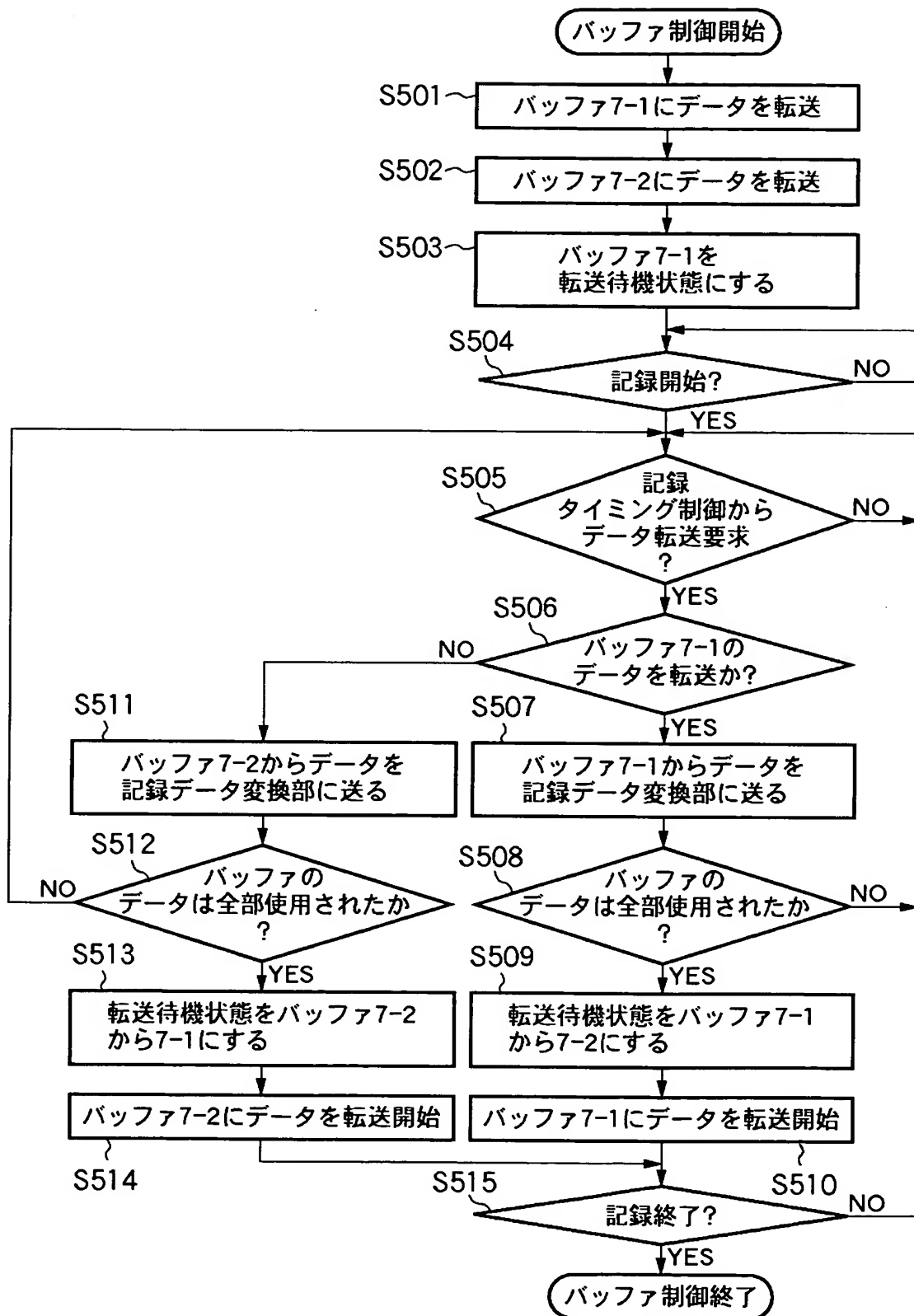
【図 3】



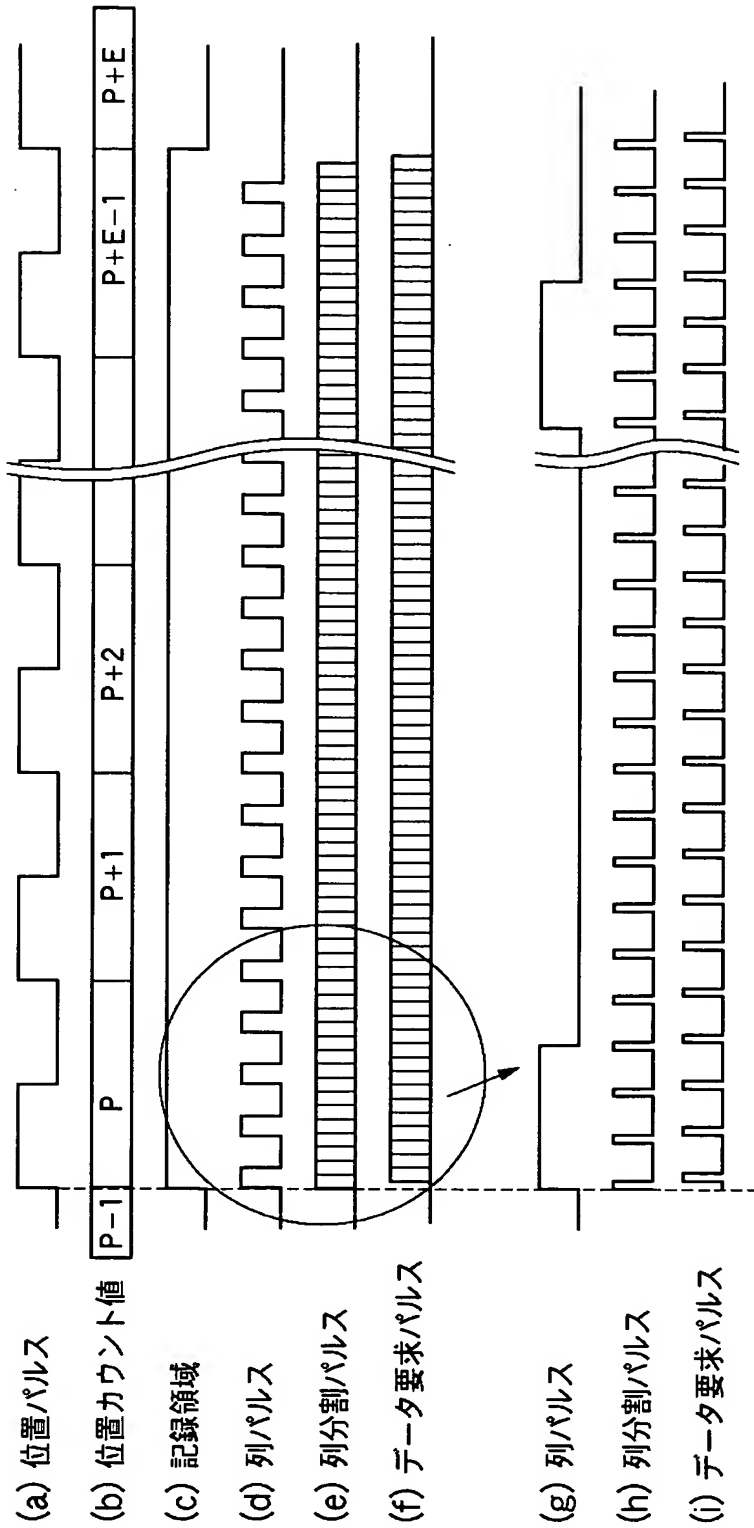
【図 4】



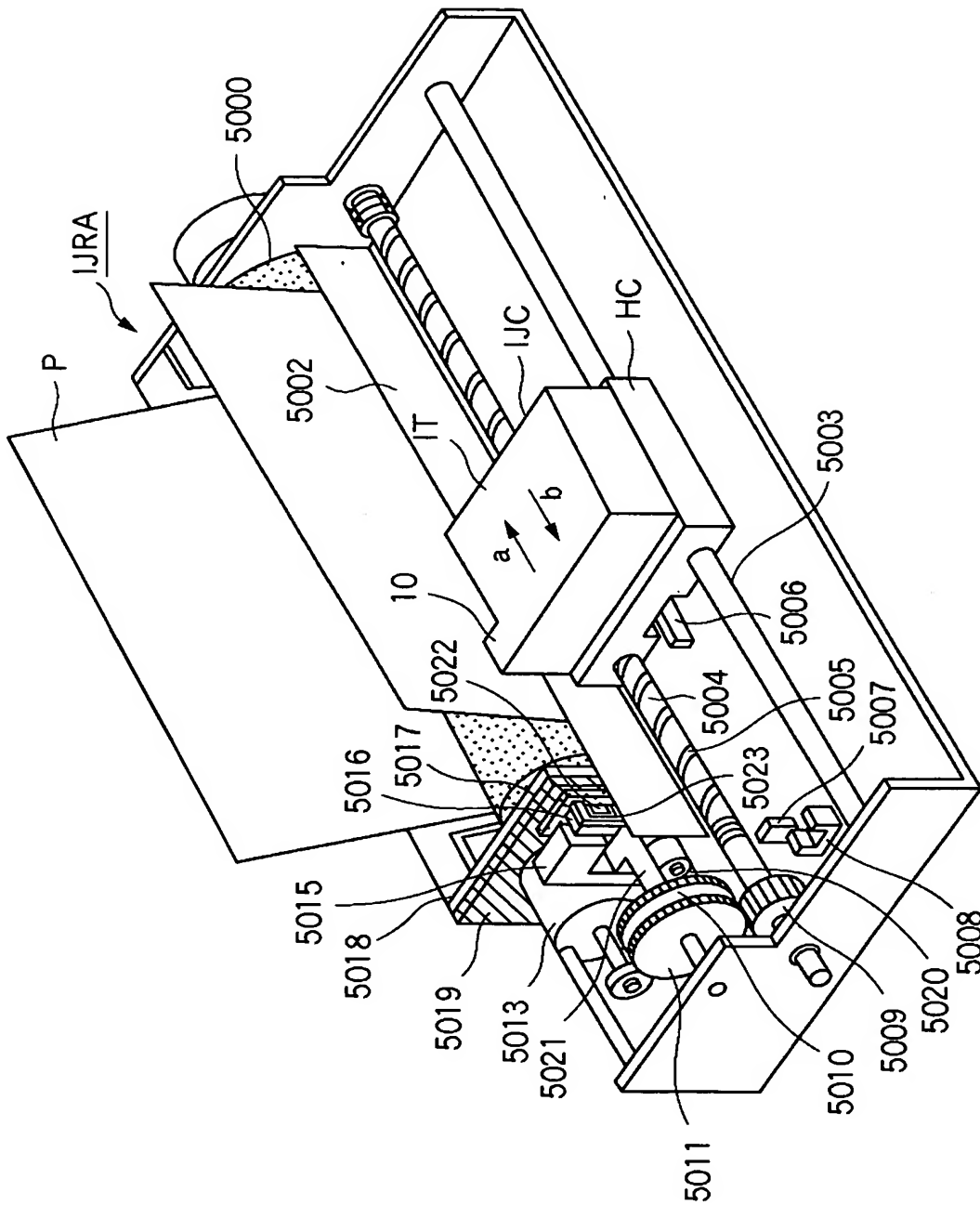
【図 5】



【図 6】



【図 7】



【図 8 A】

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity バッファ制御 is
  GENERIC(
    CONSTANT X_MSB :integer:=13;  --left range of X num-
    CONSTANT B_NUM :integer:=5;   --data num-1 to Print Cont. per block
    CONSTANT A_MSB :integer:=24;  --left range of address
    CONSTANT A_LSB :integer:=2);  --right range of address

  port(
    rst_n    :IN std_logic;        --asynchronous reset: Low active
    clk      :IN std_logic;        --Pixel clock
    c_max    :IN std_logic_vector(1 DOWNTO 0);  --color num
    c0_num   :IN std_logic_vector(8 DOWNTO 0);  --c0 nozzle num-1
    c1_num   :IN std_logic_vector(8 DOWNTO 0);  --c1 nozzle num-1
    c2_num   :IN std_logic_vector(8 DOWNTO 0);  --c2 nozzle num-1
    c3_num   :IN std_logic_vector(8 DOWNTO 0);  --c3 nozzle num-1
    blk_tbl0_0 :IN std_logic_vector(31 DOWNTO 0);  --block sequence table 0L
    blk_tbl0_1 :IN std_logic_vector(31 DOWNTO 0);  --block sequence table 0H
    n1_diff  :IN std_logic_vector(A_MSB DOWNTO A_LSB);  --next line addr diff
    c0_diff  :IN std_logic_vector(A_MSB DOWNTO A_LSB);  --c0->c1 address diff
    c1_diff  :IN std_logic_vector(A_MSB DOWNTO A_LSB);  --c1->c2 address diff
    c2_diff  :IN std_logic_vector(A_MSB DOWNTO A_LSB);  --c2->c3 address diff
    c3_diff  :IN std_logic_vector(A_MSB DOWNTO A_LSB);  --c3->c0 address diff
    bm_strt_addr :IN std_logic_vector(A_MSB DOWNTO A_LSB);  --bitmap start addr
    Bsync    :IN std_logic;        --block sync pulse
    print_e  :IN std_logic;        --print enable
    BE       :OUT std_logic_vector(3 DOWNTO 0);  --heat block
    data     :OUT std_logic_vector(B_NUM DOWNTO 0);  --heat data
    init     :IN std_logic;        --initialize from arbter
    r_na     :IN std_logic;        --Next read cycle Address request of bitmap
    r_valid  :IN std_logic;        --bitmap data valid
    r_req    :OUT std_logic;       --bitmap read request
    r_addr   :OUT std_logic_vector(A_MSB DOWNTO A_LSB);  --bitmap address
    dat_0    :IN std_logic_vector(31 DOWNTO 0);  --SRAM0 data
    dat_1    :IN std_logic_vector(31 DOWNTO 0);  --SRAM1 data
    adr_0    :OUT std_logic_vector(8 DOWNTO 0);  --SRAM0 address
    adr_1    :OUT std_logic_vector(8 DOWNTO 0);  --SRAM1 address
    we_0     :OUT std_logic;       --SRAM0 write enable
    we_1     :OUT std_logic;       --SRAM1 write enable
  );
end バッファ制御;

```


【図 8 B】

ARCHITECTURE RTL OF バッファ制御 IS

```

constant B4_0 :std_logic_vector(3 downto 0) :=(others=>'0');
constant B4_1 :std_logic_vector(3 downto 0) :=(others=>'1');
constant B5_0 :std_logic_vector(4 downto 0) :=(others=>'0');
constant B7_0 :std_logic_vector(6 downto 0) :=(others=>'0');
constant B9_0 :std_logic_vector(8 downto 0) :=(others=>'0');
constant X_11_0 :std_logic_vector(X_MSB downto 11) :=(others=>'0');
constant X_12_0 :std_logic_vector(X_MSB downto 12) :=(others=>'0');
signal ini :std_logic; --counter initialize
signal x_count :std_logic_vector(X_MSB downto 0);
signal Bsync_1 :std_logic; --block sync. delay 1
signal Bsync_2 :std_logic; --block sync. delay 2
signal Bsync_3 :std_logic; --block sync. delay 3
signal Bsync_p :std_logic; --block sync. rising edge
signal b_count :std_logic_vector(3 downto 0); --block sync. count
signal column_end :std_logic;
signal prt_ena_1 :std_logic; --print enable delay 1
signal prt_ena_2 :std_logic; --print enable delay 2
signal prt_ena_3 :std_logic; --print enable delay 3
signal Hsync_p :std_logic; --print enable rising edge
signal color_end :std_logic; --nozzle boundary of color
signal color_base :std_logic_vector(8 downto 0); --color boundary
signal buf_valid :std_logic; --buffer is valid
signal buf_valid_1 :std_logic; --buffer address is valid
signal buf_valid_2 :std_logic; --buffer output is valid
signal n_count :std_logic_vector(4 downto 0); --nozzle num in block
signal color :std_logic_vector(1 downto 0); --color num -1
signal color_1 :std_logic_vector(1 downto 0); --color for SRAM out :delay 1
signal n_num_div16 :std_logic_vector(8 downto 4); --c_num/16
signal block_end :std_logic;
signal c_num :std_logic_vector(8 downto 0); --color nozzle num-1
signal b_r_adr :std_logic_vector(8 downto 0); --buffer read address
signal r_adr_tmp :std_logic_vector(A_MSB downto A_LSB); --EXT RAM address
signal r_color :std_logic_vector(1 downto 0); --color for EXT RAM read
signal bf_chg :std_logic; --read buffer change
signal r_count :std_logic_vector(8 downto 0); --EXT RAM read counter
signal r_c_count :std_logic_vector(8 downto 0); --EXT RAM color num counter
signal r_c_end :std_logic; --EXT RAM color end
signal r_c_num :std_logic_vector(8 downto 0); --color num-1 for EXT RAM
signal r_v_count :std_logic_vector(8 downto 0); --EXT RAM read valid counter
signal r_req_tmp :std_logic; --bitmap read request
signal r_req_stack :std_logic_vector(2 downto 0); --EXT RAM read req stack
signal r_ini_end :std_logic; --1st buffer fill req end
signal w_ini_end :std_logic; --1st buffer fill write end
signal r_end :std_logic; --EXT RAM read end
signal w_end :std_logic; --buffer write end
signal data :std_logic_vector(B_NUM downto 0); --heat data
signal bn :std_logic_vector(3 downto 0); --nozzle block number
signal x_pst :std_logic_vector(X_MSB downto 0); --x position for heat data
signal bf0_rd :std_logic_vector(3 downto 0); --buffer_0 read for odd nozzle
signal bf0_rd_all :std_logic; --all nozzle read buffer_0
signal bf1_rd_all :std_logic; --all nozzle read buffer_1
signal bf0_read :std_logic; --buffer_0 read
signal bf0_read_1 :std_logic; --buffer_0 read delay 1
signal h_dat :std_logic; --heat data
signal tb_sel :std_logic_vector(1 downto 0); --block seq. table select
signal bn_seq :std_logic_vector(63 downto 0); --block sequence
signal x_diff :std_logic_vector(A_MSB downto A_LSB);
signal h_dat_s :std_logic; --heat data from SRAM
signal bf0_wite :std_logic; --buffer_0 write
signal c_max_tmp :std_logic_vector(1 downto 0); --color num for yobito
signal buf_v_ini :std_logic; --for buffer initial read
signal buf_vld_1 :std_logic; --buffer address is valid
signal pst_1 :std_logic_vector(4 downto 0); --x pos for read address

```

【図 8 C】

```

BEGIN
    ini <= NOT(print_e) OR init;
    buf_vld_1 <= buf_valid AND buf_valid_1;

    Bsync_dly : PROCESS (clk, rst_n)
    BEGIN
        IF (rst_n='0') THEN
            Bsync_1 <='0';
            Bsync_2 <='0';
            Bsync_3 <='0';
            prt_ena_1 <='0';
            prt_ena_2 <='0';
            prt_ena_3 <='0';
        ELSIF clk='1' AND clk'event THEN
            IF ini='1' OR ena='0' THEN
                Bsync_1 <='0';
                Bsync_2 <='0';
                Bsync_3 <='0';
                prt_ena_1 <='0';
                prt_ena_2 <='0';
                prt_ena_3 <='0';
            ELSE
                Bsync_1 <=Bsync;
                Bsync_2 <=Bsync_1;
                Bsync_3 <=Bsync_2;
                prt_ena_1 <=print_e;
                prt_ena_2 <=prt_ena_1;
                prt_ena_3 <=prt_ena_2;
            END IF;
        END IF;
    END IF;
    END PROCESS Bsync_dly;

    Bsync_pulse : PROCESS (Bsync_2, Bsync_3)
    BEGIN
        IF Bsync_2='1' AND Bsync_3='0' THEN
            Bsync_p <='1';
        ELSE
            Bsync_p <='0';
        END IF;
    END PROCESS Bsync_pulse;

    Hsync_pulse : PROCESS (prt_ena_2, prt_ena_3)
    BEGIN
        IF prt_ena_2='1' AND prt_ena_3='0' THEN
            Hsync_p <='1';
        ELSE
            Hsync_p <='0';
        END IF;
    END PROCESS Hsync_pulse;

--for buffer read
    dr_bas_gen : PROCESS (clk, rst_n)
    BEGIN
        IF (rst_n='0') THEN
            color_base <=(others =>'0');
        ELSIF clk='1' AND clk'event THEN
            IF init='1' OR block_end='1' THEN
                color_base <=(others =>'0');
            ELSIF color_end='1' THEN
                color_base <=color_base+c_num+'1';
            END IF;
        END IF;
    END PROCESS dr_bas_gen;

```

801

【図 8 D】

```

num_cnt : PROCESS(dk, rst_n)--color access counter of each block
BEGIN
  IF (rst_n='0') THEN
    n_count<=(others=>'0');
  ELSIF clk='1' AND clk'event THEN
    IF buf_vid_1='0' OR color_end='1' THEN
      n_count<=(others=>'0');
    ELSE
      n_count<=n_count+'1';
    END IF;
  END IF;
END PROCESS num_cnt;

end_ncnt_det : PROCESS (n_count, n_num_div16, buf_vid_1)
BEGIN
  IF n_count=n_num_div16 THEN
    color_end<=buf_vid_1;
  ELSE
    color_end<='0';
  END IF;
END PROCESS end_ncnt_det;

color_cnt : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    color<=(others=>'0');
  ELSIF clk='1' AND clk'event THEN
    IF init='1' OR block_end='1' THEN
      color<=(others=>'0');
    ELSIF color_end='1' THEN
      color<=color+'1';
    END IF;
  END IF;
END PROCESS color_cnt;

end_blk_det : PROCESS (color, c_max_tmp, color_end)
BEGIN
  IF color=c_max_tmp THEN
    block_end<=color_end;
  ELSE
    block_end<='0';
  END IF;
END PROCESS end_blk_det;

div16_mux : PROCESS (color, c0_num, c1_num, c2_num, c3_num, c_num)
BEGIN
  CASE color IS
    WHEN '00' => c_num<=c0_num;
    WHEN '01' => c_num<=c1_num;
    WHEN '10' => c_num<=c2_num;
    WHEN others => c_num<=c3_num;
  END CASE;
  n_num_div16<=c_num(8 DOWNTO 4);
END PROCESS div16_mux;

buf_vid_gen : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    buf_valid<='0';
  ELSIF clk='1' AND clk'event THEN
    IF init='1' OR block_end='1' THEN
      buf_valid<='0';
    ELSIF Bsync_p='1' THEN
      buf_valid<='1';
    ELSE
      IF w_ini_end='0' AND w_end='1' THEN
        buf_valid<='1';
      END IF;
    END IF;
  END IF;
END PROCESS buf_vid_gen;

```

【図 8 E】

```

b_cnt : PROCESS (rst_n, clk)
BEGIN
    IF rst_n='0' THEN
        b_count<=(others =>'0');
    ELSIF clk='1' AND clk'event THEN
        IF ini='1' OR Hsync_p='1' OR column_end='1' THEN
            b_count<=(others =>'0');
        ELSIF Bsync_p='1' THEN
            b_count<=b_count+'1';          --count up
        END IF;
    END IF;
END PROCESS b_cnt;

end_Block_det : PROCESS (dir, bs_auto, b_count, Bsync_p)
BEGIN
    IF dir='0' OR bs_auto='0' THEN
        IF b_count=B4_1 THEN
            column_end<=Bsync_p;
        ELSE
            column_end<='0';
        END IF;
    ELSE
        IF b_count=B4_0 THEN
            column_end<=Bsync_p;
        ELSE
            column_end<='0';
        END IF;
    END IF;
END PROCESS end_Block_det;

x_cnt : PROCESS (rst_n, clk)
BEGIN
    IF rst_n='0' THEN
        x_count<=(others =>'0');
    ELSIF clk='1' AND clk'event THEN
        IF ini='1' OR Hsync_p='1' THEN
            x_count<=(others =>'0');
        ELSIF column_end='1' THEN
            x_count<=x_count+'1';
        END IF;
    END IF;
END PROCESS x_cnt;

x_pst<=x_count+x_off;

b_r_adr_gen : PROCESS (n_count, color_base, bn, x_pst)
variable  b_r_adr_tmp          : std_logic_vector(8 downto 0);
BEGIN
    b_r_adr_tmp :=color_base+(n_count(4 downto 1)& bn & n_count(0));
    b_r_adr<=b_r_adr_tmp(7 downto 0)& x_pst(5);
END PROCESS b_r_adr_gen;

```

【図 8 F】

```

bf_vld_dly : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    buf_valid_1<='0';
    buf_valid_2<='0';
  ELSIF clk='1' AND clk'event THEN
    IF init='1' THEN
      buf_valid_1<='0';
      buf_valid_2<='0';
    ELSE
      buf_valid_1<=buf_valid;
      buf_valid_2<=buf_vld_1;
    END IF;
  END IF;
END PROCESS bf_vld_dly;

color_dly : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    color_1<=(others=>'0');
  ELSIF clk='1' AND clk'event THEN
    color_1<=color;
  END IF;
END PROCESS color_dly;

bf0_read<=NOT x_pst(5);

bf0_read_dly : PROCESS (rst_n, clk)
BEGIN
  IF rst_n='0' THEN
    bf0_read_1<='0';
  ELSIF clk='1' AND clk'event THEN
    bf0_read_1<=bf0_read;
  END IF;
END PROCESS bf0_read_dly;

bn_seq<=blk_tbl0_0 & blk_tbl0_1;
bn_dly : PROCESS (rst_n, clk)
variable  bn_sel      : std_logic_vector(3 downto 0);
variable  lsb         : integer range 0 to 63;
BEGIN
  IF rst_n='0' THEN
    bn<=(others=>'0');
  ELSIF clk='1' AND clk'event THEN
    IF buf_valid='1' AND buf_valid_1='0' THEN
      bn_sel :=NOT (b_count);
      lsb :=CONV_INTEGER ('0' & bn_sel & '00');
      FOR i IN 0 TO 3 LOOP
        bn(i)<=bn_seq(i+lsb);
      END LOOP;
    END IF;
  END IF;
END PROCESS bn_dly;

```

【図 8 G】

```

bf0_rd_d : PROCESS (rst_n, clk)
BEGIN
    IF rst_n='0' THEN
        bf0_rd<= (others =>'0');
    ELSIF clk='1' AND clk'event THEN
        IF init='1' THEN
            bf0_rd<= (others =>'0');
        ELSIF buf_vid_1='1' THEN
            bf0_rd (CONV_INTEGER(color))<=bf0_read;
        END IF;
    END IF;
END PROCESS bf0_rd_d;

bf_rd_all_det : PROCESS (c_max_tmp, e_bf0_rd, o_bf0_rd)
BEGIN
    bf0_rd_all<='0';
    bf1_rd_all<='0';
    CASE c_max_tmp IS
        WHEN "00" =>
            IF bf0_rd(0)='1' THEN
                bf0_rd_all<='1';
            END IF;
            IF bf0_rd(0)='0' THEN
                bf1_rd_all<='1';
            END IF;
        WHEN "01" =>
            IF bf0_rd(1 DOWNTO 0)="11" THEN
                bf0_rd_all<='1';
            END IF;
            IF bf0_rd(1 DOWNTO 0)="00" THEN
                bf1_rd_all<='1';
            END IF;
        WHEN "10" =>
            IF bf0_rd(2 DOWNTO 0)="111" THEN
                bf0_rd_all<='1';
            END IF;
            IF bf0_rd(2 DOWNTO 0)="000" THEN
                bf1_rd_all<='1';
            END IF;
        WHEN others =>
            IF bf0_rd(3 DOWNTO 0)="1111" THEN
                bf0_rd_all<='1';
            END IF;
            IF bf0_rd(3 DOWNTO 0)="0000" THEN
                bf1_rd_all<='1';
            END IF;
    END CASE;
END PROCESS bf_rd_all_det;

bf_write_det : PROCESS (rst_n, clk)
BEGIN
    IF rst_n='0' THEN
        bf0_wite<='1';
    ELSIF clk='1' AND clk'event THEN
        IF init='1' THEN
            bf0_wite<='1';
        ELSIF bf_chg_1='1' THEN
            bf0_wite<=NOT bf0_wite;
        END IF;
    END IF;
END PROCESS bf_write_det;

bf_chg_gen : PROCESS (Hsync_p, w_ini_end, w_end, bf0_wite, bf0_rd_all, bf1_rd_all)
variable ini_chg : std_logic;
BEGIN
    ini_chg := (NOT w_ini_end AND w_end);
    IF bf0_wite='1' THEN
        bf_chg<=ini_chg OR (w_end AND bf0_rd_all);
    ELSE
        bf_chg<=ini_chg OR (w_end AND bf1_rd_all);
    END IF;
END PROCESS bf_chg_gen;

```

【図 8 H】

```
pst_dly : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    pst_1<=(others =>'1');
  ELSIF clk='1' AND clk'event THEN
    pst_1<=NOT x_pst(4 downto 0);
  END IF;
END PROCESS pst_dly;

h_dat_s_mux : PROCESS (pst_1, bf0_read_1, dat_0, dat_1)
variable   pst           : integer range 0 to 31;
BEGIN
  pst := CONV_INTEGER ('0' & pst_1);

  IF bf0_read_1='1' THEN
    h_dat_s<=dat_0(pst);
  ELSE
    h_dat_s<=dat_1(pst);
  END IF;
END PROCESS h_dat_s_mux;

h_dat<=h_dat_s;

out_pack : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    data<=(others =>'0');
  ELSIF clk='1' AND clk'event THEN
    IF buf_valid_2='1' AND (print_e='1' OR buf_v_ini='1') THEN
      data(0)<=h_dat;
      FOR i IN TO B_NUM LOOP
        data(i)<=data(i-1);
      END LOOP;
    END IF;
  END IF;
END PROCESS out_pack;
```

【図 8 I】

```

r_v_cnt : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    r_v_count<=(others =>'0');           --buffer write address
  ELSIF clk='1' AND clk'event THEN
    IF init='1' OR bf_chg='1' THEN
      r_v_count<=(others =>'0');
    ELSIF r_valid='1' THEN
      r_v_count<=r_v_count+'1';
    END IF;
  END IF;
END PROCESS r_v_cnt;

end_w_ini : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    w_ini_end<='0';
  ELSIF clk='1' AND clk'event THEN
    IF init='1' THEN
      w_ini_end<='0';
    ELSIF w_end='1' THEN
      w_ini_end<='1';
    END IF;
  END IF;
END PROCESS end_w_ini;

w_end_det : PROCESS (r_req_tmp, r_req_stack)
BEGIN
  IF r_req_tmp='0' AND r_req_stack="000" THEN
    w_end<='1';
  ELSE
    w_end<='0';
  END IF;
END PROCESS w_end_det;

bf_adr_mux : PROCESS (r_valid, bf0_wite, print_e, yp_yobi, b_r_adr, r_v_count)
BEGIN
  IF r_valid='1' THEN
    IF bf0_wite='0' THEN
      adr_0<=b_r_adr;
      adr_1<=r_v_count;
      we_0<='0';
      we_1<='1';
    ELSE
      adr_0<=r_v_count;
      adr_1<=b_r_adr;
      we_0<='1';
      we_1<='0';
    END IF;
  ELSE
    we_0<='0';
    we_1<='0';
    adr_0<=b_r_adr;
    adr_1<=b_r_adr;
  END IF;
END PROCESS bf_adr_mux;

bf_v_ini_det : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    buf_v_ini<='0';
  ELSIF clk='1' AND clk'event THEN
    IF init='1' OR block_end='1' THEN
      buf_v_ini<='0';
    ELSE
      IF w_ini_end='0' AND w_end='1' THEN
        buf_v_ini<='1';
      END IF;
    END IF;
  END IF;
END PROCESS bf_v_ini_det;

```


【図 8 J】

```

cmax_det : PROCESS (c_max)
BEGIN
    IF c_max="11" THEN
        c_max_tmp<="11";
    ELSE
        c_max_tmp<=c_max;
    END IF;
END PROCESS cmax_det;

r_req_gen : PROCESS (clk, rst_n)
BEGIN
    IF (rst_n='0') THEN
        r_req_tmp<='0';
    ELSIF clk='1' AND clk'event THEN
        IF init='1' OR bf_chg='1' THEN
            r_req_tmp<='1';
        ELSIF r_end='1' THEN
            r_req_tmp<='0';
        END IF;
    END IF;
END PROCESS r_req_gen;

r_req_stck : PROCESS (clk, rst_n)
BEGIN
    IF (rst_n='0') THEN
        r_req_stck<=(others =>'1');
    ELSIF clk='1' AND clk'event THEN
        IF init='1' THEN
            r_req_stck<=(others =>'0');
        ELSIF r_na='1' THEN
            IF r_valid='0' THEN
                r_req_stck(2 downto 1)<=r_req_stck(2 downto 1)+'1';
            ELSE
                r_req_stck<=r_req_stck+'1';
            END IF;
        ELSIF r_valid='1' THEN
            r_req_stck<=r_req_stck-'1';
        END IF;
    END IF;
END PROCESS r_req_stck;

end_r_ini : PROCESS (clk, rst_n)
BEGIN
    IF (rst_n='0') THEN
        r_ini_end<='0';
    ELSIF clk='1' AND clk'event THEN
        IF init='1' THEN
            r_ini_end<='0';
        ELSIF r_end='1' THEN
            r_ini_end<='1';
        END IF;
    END IF;
END PROCESS end_r_ini;

r_end_det : PROCESS (r_color, c_max_tmp, r_c_end)
BEGIN
    IF r_color=c_max_tmp AND r_c_end='1' THEN
        r_end<='1';
    ELSE
        r_end<='0';
    END IF;
END PROCESS r_end_det;

r_adr<=r_adr_tmp;
r_req<=r_req_tmp;
BE<=o_bn;

```

【図 8 K】

```

--for EXT RAM read
r_ad_gen : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    r_adr_tmp<=(others =>'0');
  ELSIF clk='1' AND clk'event THEN
    IF init='1' THEN
      r_adr_tmp<=bm_strt_adr;
    ELSIF r_na='1' THEN
      IF r_c_end='1' THEN
        r_adr_tmp<=r_adr_tmp+x_diff;
      ELSE
        r_adr_tmp<=r_adr_tmp+n1_diff;
      END IF;
    END IF;
  END IF;
END PROCESS r_ad_gen;

df_sel : PROCESS (r_color, c0_diff, c1_diff, c2_diff, c3_diff, c0_num, c1_num, c2_num, c3_num)
BEGIN
  CASE r_color IS
    WHEN "00" => x_diff<=c0_diff; r_c_num<=c0_num;
    WHEN "01" => x_diff<=c1_diff; r_c_num<=c1_num;
    WHEN "10" => x_diff<=c2_diff; r_c_num<=c2_num;
    WHEN others => x_diff<=c3_diff; r_c_num<=c3_num;
  END CASE;
END PROCESS df_sel;

c_cnt : PROCESS (clk, rst_n) --EXT RAM read counter for color
BEGIN
  IF (rst_n='0') THEN
    r_c_count<=(others =>'0');
  ELSIF clk='1' AND clk'event THEN
    IF init='1' THEN
      r_c_count<=(others =>'0');
    ELSIF r_c_end='1' THEN
      r_c_count<=(others =>'0');
    ELSIF r_na='1' THEN
      r_c_count<=r_c_count+'1';
    END IF;
  END IF;
END PROCESS c_cnt;

r_col_cnt : PROCESS (clk, rst_n)
BEGIN
  IF (rst_n='0') THEN
    r_color<=(others =>'0');
  ELSIF clk='1' AND clk'event THEN
    IF init='1' OR bf_chg='1' THEN
      r_color<=(others =>'0');
    ELSIF r_c_end='1' THEN
      r_color<=r_color+'1';
    END IF;
  END IF;
END PROCESS r_col_cnt;

end_col_det : PROCESS (r_c_count, r_c_num, r_na)
BEGIN
  IF r_count=r_c_num THEN
    r_c_end<=r_na;
  ELSE
    r_c_end<='0';
  END IF;
END PROCESS end_col_det;

END RTL;

```

【図 9】

Parameter Name	Parameter Address	Parameter Value
c_num	0(Hex)	2
c0_n_num	4(Hex)	n
c1_n_num	8(Hex)	m
c2_n_num	C(Hex)	l
c0_diff	10(Hex)	1
c1_diff	14(Hex)	1
c2_diff	18(Hex)	FFFFFFC1
bm_strt_adr	1C(Hex)	100
nl_diff	20(Hex)	40

【書類名】 要約書

【要約】

【課題】 様々な構成の記録ヘッドに対して共通の制御回路を使用する。

【解決手段】 所定方向に配列された複数の記録素子を有する記録ヘッドを搭載したキャリッジを、記録素子の配列方向と交差する方向に記録媒体上で走査させて記録を行う記録装置において、ラスタ形式の記録データを記録データメモリ 6 に格納し、記録データメモリ 6 に格納された記録データを各記録素子に対応して格納領域を有するバッファメモリ 7 に格納する構成において、記録ヘッドの構成に関する情報を格納するヘッドパラメータ部 4 を設け、ヘッドパラメータ部 4 に格納された情報に応じて、記録データメモリ 6 に格納された記録データのバッファメモリ 7 への転送順序と、バッファメモリ 7 に格納された記録データの読み出し順序とをバッファ制御部 2 及び記録データ用メモリ読み出し制御部 11 で制御する。

【選択図】 図 2

特願 2 0 0 3 - 0 8 1 0 6 0

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 1 0 0 7]

1. 変更年月日

1 9 9 0 年 8 月 3 0 日

[変更理由]

新規登録

住 所

東京都大田区下丸子 3 丁目 3 0 番 2 号

氏 名

キヤノン株式会社